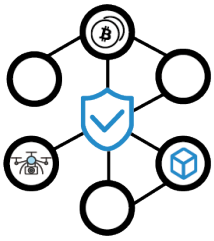

苏州大学网络空间安全实验室

Cyberspace Security Lab of Soochow University

组会

组会汇报标题

姓　　名：叶冬冬

汇 报 心 情：良　好

汇报身体状况：健　康

2020 年 7 月

摘　要

这份模板是苏州大学网络空间安全实验室所使用的报告模板。我从很久以前就在 overleaf 上搜索苏州大学的 LaTeX 模板，发现总是一无所获。

我希望所有的报告都有一份模板，所有人都能方便快捷地组织自己的文章，再也不用为格式所困扰。本科期间我自己从来不会因格式而困扰，因为我对这些东西颇有研究。但是，往往会有很多同学突然拿着他的论文就过来了，哪哪页码不会设置，哪哪莫名其妙多出空行，我往往要花费大量时间教他们如何修改，更可气的是，很多时候大家的问题都是相同的，同样的话我要说无数遍。更别提他们，在老师没有说格式要求时，写的不是人看的东西了。

我希望我们实验室有一套自己的工作流程，无论工作做得好不好，态度总是要有的。用这句话结尾吧：真正的光明决不是用没有黑暗的时间，只是用不被黑暗的时间所遮蔽；真正的英雄决不是用没有卑下的情操，只是永不被卑下的情操所屈服罢了。[①]

关 键 词：苏州大学，网络安全实验室，态度

论文类型：应用基础

[①] 严复为《约翰·克里斯朵夫》题词

ABSTRACT

You will never want to use Word when you have learned how to use LaTeX.

KEY WORDS: LaTeX, TexLive2016, XeLaTex

TYPE OF DISSERTATION: Application Fundamentals

目 录

1 相机校正模型与方法

1.1 建立相机成像的几何模型

得到物体从三位世界映射到相机成像平面的变换矩阵，这一过程最关键的部分就是需要得到相机的内参和外参。

1.1.1 坐标系概念

世界坐标系 (world coordinate system) 是用户定义的三维世界的坐标系，为了描述目标物在真实世界里的位置而被引入，坐标 (x_w, y_w, z_w)，单位为 m。相机坐标 [1] 系 (camera coordinate system), 在相机上建立的坐标系，为了从相机的角度描述物体位置而定义，作为沟通世界坐标系和图像/像素坐标系的中间一环，坐标 (x_c, y_c, z_c)，单位为 m。图像坐标系 (image coordinate system), 也有看到称作“像平面坐标系”的，为了描述成像过程中物体从相机坐标系到图像坐标系的投影透射关系而引入，方便进一步得到像素坐标系下的坐标，坐标 (x, y)，单位为 m。像素坐标系 (pixel coordinate system)，为了描述物体成像后的像点在数字图像上的坐标而引入，是我们真正从相机内读取到的信息所在的坐标系，坐标 (u, v)，单位为 $pixels$（像素数目）。图 1-1 表示的是以上这三种坐标系的示意图

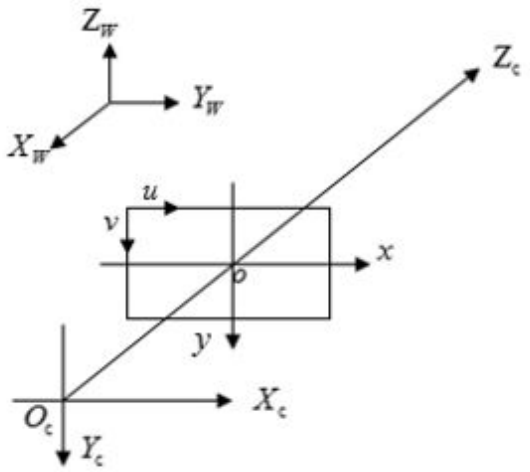

图 1-1 三种坐标系

1.1.2 世界坐标系到相机坐标系

刚体从世界坐标系转换到相机坐标系的过程，可以通过旋转和平移来得到，其变换矩阵由一个旋转矩阵 R_{3x3} , 和平移向量 T_{3x1} 组合而成, 如图 1-2 所示 可以由式 (1-1) 推导出 P 在相机坐标系中的坐标，如式 (1-2) ，其中变换矩阵为外参矩阵

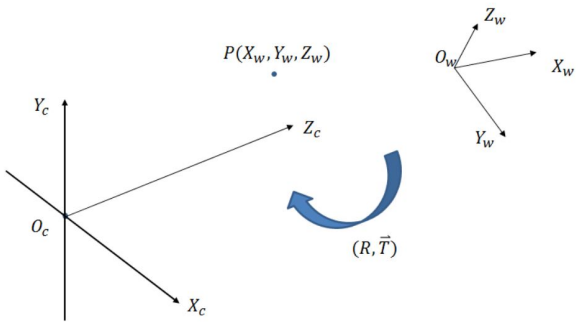

图 1-2 世界坐标系到相机坐标系

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \tag{1-1a}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ \bar{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1-2a}$$

1.1.3 相机坐标系到图像坐标系

从相机坐标系到图像坐标系，属于透视投影关系，从 3D 转换到 2D。也就是把三维物体成像到二维成像面的过程，一般用简化的成像模型——小孔成像模型，这种情况下图像一定会落在焦平面上。但实际光学系统中是透镜成像，一般物体的像都不会落在焦平面，成像模型比较复杂。如图 1-3 ，可以推导出公式 1-3

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{1-3a}$$

1.1.4 图像坐标系到像素坐标系

由于定义的像素坐标系原点与图像坐标系原点不重合，假设图像坐标系原点 o 在像素坐标系 O_{uv} 下的坐标为 $(u0, v0)$，每个像素点在图像坐标系 x 轴、y 轴方向的尺寸为：dx , dy（这部分在实际硬件中为成像元件 CCD 或者 CMOS 中成像单元的中心间距，由成像单元的大小和间距决定），且像点 p 在实际图像坐标系下的坐标为 (x, y) ，如图 1-4 ，可得到像点 p 在像素坐标系 O_{uv} 下的坐标 (u, v) 与前面方法相似，可以推

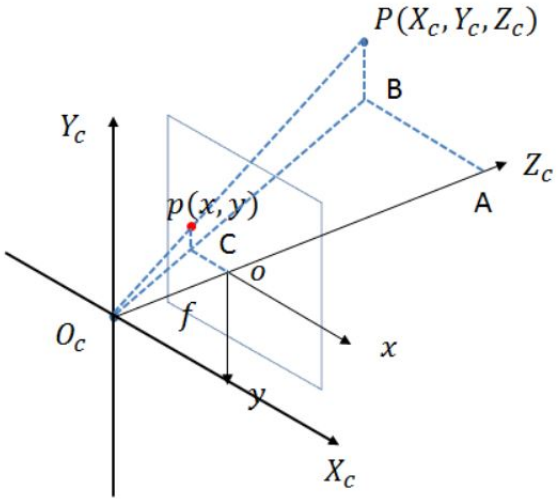

图 1-3 小孔成像模型

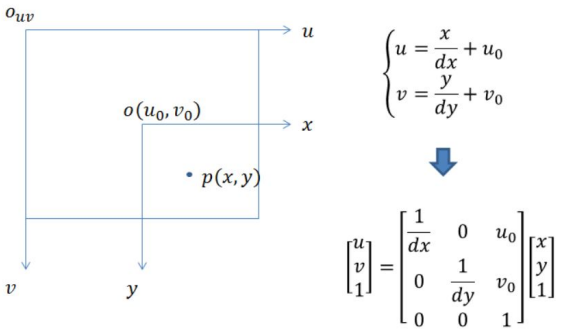

图 1-4 图像坐标系到像素坐标系

导出内参矩阵如式 1-4

$$\begin{bmatrix} f/d_x & 0 & u_0 \\ 0 & f/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1-4a}$$

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/d_x & 0 & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ \bar{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ \bar{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1-5a}$$

1.2 相机的畸变模型

理论上讲是可能定义一种透镜而不引入任何畸变的，比如说之前建立的模型，但是现实世界并没有那么完美的透镜。这主要是制造上的原因，因为制作一个“球形”透镜

比制作一个数学上理想的透镜更加容易，而且从机械方面很难把透镜和成像仪保持平行。这里主要描述两种主要的透镜畸变并且建立其模型。径向畸变来自于透镜的形状，而切向畸变则来自于整个摄像机的组装过程。

实际摄像机的透镜总是在成像仪的边缘产生显著的畸变，这个现象源于“鱼眼”影响，径向畸变如图 1-5 所示，远离透镜中心的光线弯曲比较靠近中心的严重，因此正方形的边在图像平面上为弯曲（即鱼眼畸变）。对于便宜的网络摄像头，可以用泰勒级数展开的前几项来定量描述，通常使用前三项 k_1，k_2 ，k_3 来描述，如式子 1.3.1

$$\begin{cases} x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^4) \\ y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^4) \end{cases} \tag{1-6a}$$

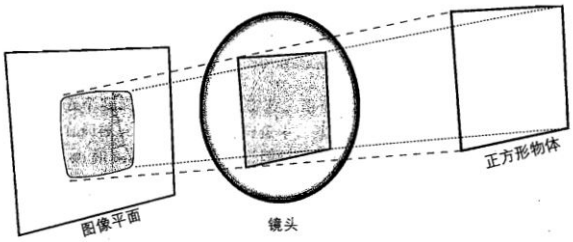

图 1-5 径向畸变

而如图 1-6 所示，是切向畸变，可以用两个额外的参数 P_1，P_2 来描述, 如式子 1-7

$$\begin{cases} x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)] \\ y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 x] \end{cases} \tag{1-7a}$$

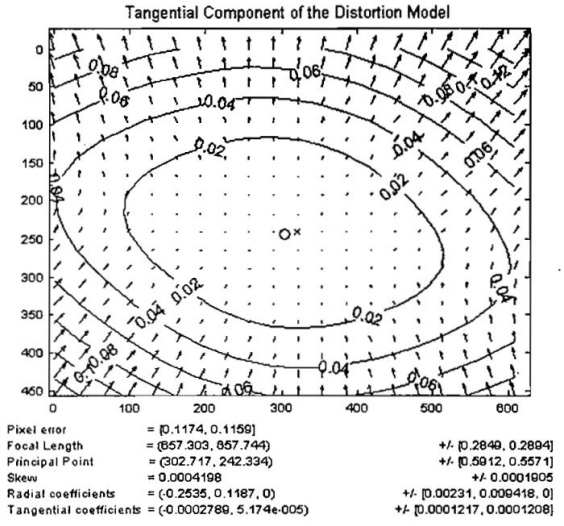

图 1-6 切向畸变

1.3 张正友标定算法原理

“张正友标定”是相机标定领域的杰出研究成果，是张正友教授 1998 年提出的单平面棋盘格的相机标定方法。文中提出的方法介于传统标定法和自标定法之间，但克服了传统标定法需要的高精度标定物的缺点，仅仅需要一个打印出来的棋盘格就可以了，同时也相对于自标定而言，提高了精度，便于操作。因此，张正友标定法被广泛应用于计算机视觉方面。

1.3.1 计算外参

设三维世界坐标的点为 $M=[X,Y,Z,1]T$,二维相机平面像素坐标为 $m=[u,v,1]T$，所以如上所述，可以知道标定用的棋盘格平面到图像平面的单应性关系为：$sm = A[R,t]M$。其中 s ： 世界坐标系到图像坐标系的尺度因子 A：相机内参矩阵，参考式，A 多了个径向畸变参数 γ (u_0,v_0) 像主点坐标 α，β ： 焦距与像素横纵比的融合 γ ： 径向畸变参数 R ： 旋转矩阵 t ： 平移向量

$$A=\begin{bmatrix}\alpha & \gamma & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix} \tag{1-8a}$$

不妨设棋盘格位于 $Z=0$，定义旋转矩阵 R 的第 i 列为 r_i, 则有：

$$s\begin{bmatrix}u\\ v\\ 1\end{bmatrix}=A\begin{bmatrix}r_1 & r_2 & r_3 & 1\end{bmatrix}\begin{bmatrix}X\\ Y\\ 0\\ 1\end{bmatrix}=A\begin{bmatrix}r_1 & r_2 & t\end{bmatrix}\begin{bmatrix}X\\ Y\\ 1\end{bmatrix} \tag{1-9a}$$

令 $H=[h_1h_2h_3]=A[r_1r_2t]$, 于是空间到图像的映射可改为：$sm=HM$，其中 H 是描述 $Homographic$ 矩阵，H 是一个齐次矩阵，所以有 8 个未知数，至少需要 8 个方程，每对对应点能提供两个方程，所以至少需要四个对应点，就可以算出世界平面到图像平面的单应性矩阵 H。因此可以推导出：

$$r_1=\lambda A^{-1}h_1$$

$$r_2=\lambda A^{-1}h_2$$

$$r_3=r_1*r_2$$

$$t=\lambda A^{-1}h_3$$

$$\lambda=1/||A^{-1}h_1||=1/||A^{-1}h_2||$$

1.3.2 计算内参

一般而言, 求解出的 R 不会满足正交与归一化的标准, 实际操作中, 需要对 R 进行 SVD 分解。推到略, 可以得到 r_1 和 r_2 正交, 并且模相等, 可以得到约 $h_1^T A^{-T} A^{-T} h_2 = 0$, $h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2$ 。定义

$$B = A^{-T} A^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \tag{1-10a}$$

B 中的未知量可以表示成 6D 向量 $\vec{b}$, $\vec{b} = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}]^T$, 设 H 中的第 i 列为 h_i , $h_i = [h_{i1} \quad h_{i2} \quad h_{i3}]$ 可以推到出 $h_i^T B h_j = v_{ij}^T \vec{b}$。可以推导出:

$$B = A^{-T} A^{-1} = \begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} \vec{b} = 0 \tag{1-11a}$$

根据推到的结果可知如果有 n 组观察图像, 则 V 是 2n x 6 的矩阵。根据最小二乘定义, $vb = 0$ 的解是 $v^T v$ 最小特征值对应的特征向量。因此, 可以直接估算出 $\vec{b}$, 后续可以通过 $\vec{b}$ 求解内参因为 B 中的未知量为 6 个, 所以当观测平面 n 3 时, 可以得到 $\vec{b}$ 的唯一解当 n = 2 时, 一般可令畸变参数 $\gamma = 0$; 当 n = 1 时, 仅能估算出 α 与 β, 此时一般可假定像主点坐标 u_0 与 v_0 为 0。内部参数可通过如下公式计算 (cholesky 分解):

$$v_0 = (B_{12} B_{13} - B_{11} B_{23}) / (B_{11} B_{22} - B_1 2^2)$$

$$\lambda = B_{33} - [B_{13}^2 + v_0 (B_{12} B_{13} - B_{11} B_{23})] / B_{11}$$

$$\alpha = \sqrt{\lambda / B_{11}}$$

$$\beta = \sqrt{\lambda B_{11} / (B_{11} B_{22} - B_{12}^2)}$$

$$\gamma = -B_{12} \alpha^2 \beta / \lambda$$

$$u_0 = \gamma v_0 / \alpha - B_1 3 \alpha^2 / \lambda$$

1.3.3 最大似然估计

上述的推导结果是基于理想情况下的解, 但由于可能存在高斯噪声, 所以使用最大似然估计进行优化。设我们采集了 n 副包含棋盘格的图像进行定标, 每个图像里有棋盘格角点 m 个。令第 i 副图像上的角点 M_j 在上述计算得到的摄像机矩阵下图像上的投影点为:

$$\hat{m}(K, R_I, t_i, M_{ij}) = K[R|t] M_{ij}$$

其中 R_i 和 t_i 是第 i 副图对应的旋转矩阵和平移向量，K 是内参数矩阵。则角点 m_{ij} 的概率密度函数为：

$$f(m_{ij}) = 1\frac{1}{\sqrt{2\pi}}e^{\frac{-(\hat{m}(K_i,R_i,t_i,M_{i,j})-m_{ij})^2}{\delta^2}}$$

构造似然函数：

$$L(A,R_i,t_i,M_{ij}) = \prod_{i=1,j=1}^{n,m} f(m_{ij}) = 1\frac{1}{\sqrt{2\pi}}e^{\frac{-(\sum_{i=1}^{n}\sum_{j=1}^{m}||\hat{m}(K_i,R_i,t_i,M_{i,j})||^2}{\delta^2}}$$

让 L 取得最大值，即让下面式子最小。这里使用的是多参数非线性系统优化问题的 Levenberg-Marquardt 算法进行迭代求最优解。

$$\sum_{i=1}^{n}\sum_{j=1}^{m}||\hat{m}(K_i,R_i,t_i,M_{i,j})||^2$$

1.3.4 径向畸变估计

张氏标定法只关注了影响最大的径向畸变。则数学表达式为 1-12：

$$\begin{cases}\hat{u} = u + (u-u_0)[k_1(x^2+y^2)+k_2(x^2+y^2)^2] \\ \hat{v} = v + (v-v_0)[k_1(x^2+y^2)+k_2(x^2+y^2)^2]\end{cases} \tag{1-12a}$$

其中，(u,v) 是理想无畸变的像素坐标，$(\hat{u},\hat{v})$ 是实际畸变后的像素坐标。(u_0,v_0) 代表主点，(x,y) 是理想无畸变的连续图像坐标，$(\hat{x},\hat{y})$ 是实际畸变后的连续图像坐标。k_1 和 k_2 为前两阶的畸变参数。

$$\hat{u} = u_0 + \alpha\hat{x} + \gamma\hat{y}$$

$$\hat{v} = v_0 + \beta\hat{y}$$

化作矩阵形式 1-13：

$$\begin{bmatrix}(u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2\end{bmatrix}\begin{bmatrix}k_1 \\ k_2\end{bmatrix} = \begin{bmatrix}\hat{u}-u \\ \hat{v}-v\end{bmatrix} \tag{1-13a}$$

记作：$D_k = d$ 可得到畸变系数 k：

$$k = [k1 \quad k_2]^T = (D^TD)^{-1}D^Td$$

使用最大似然的思想优化得到的结果，即像上一步一样，LM 法计算下列函数值最小的参数值：

$$\sum_{i=1}^{n}\sum_{j=1}^{m}||\hat{m}(K_i,k_1,k_2,R_i,t_i,M_{i,j})||^2$$

表 1-1 表题也是五号字

Interference	DOA (deg)	Bandwidth (MHz)	INR (dB)
1	−30	20	60
2	20	10	50
3	40	5	40

至此，张氏标定法介绍完毕。得到了相机内参、外参和畸变系数。表格要求采用三线表，如表 1-1 所示。

1.4 算法

1.4.1 单个算法

Data: 计算节点 client
Result: 计算文件 `test.py`

```
1  begin
2      data = client.recv(1024)
3      if data == b'exist' then
4          if os.path.exists("test.py") then
5              client.send("Y".encode('utf-8'))
6          else
7              client.send("N".encode('utf-8'))
8              while True do
9                  with open("test.py", "ab") as f
10                 data = client.recv(1024)
11                 if data == b'quit' then
12                     client.send("received".encode('utf-8'))
13                     break
14                 else
15                 end
16                 f.write(data)
17             end
18             print(" 节点%d：计算文件已经接收！存储为 test.py" % (cnt)) break
19         end
20     else
21     end
22 end
```

算法 1-1 FileSplitSend()

2 实验结果与分析

2.1 相机标定步骤

实际操作和标定的代码中，一般都遵循如下步骤：1、打印一张棋盘格，把它贴在一个平面上，作为标定物。
2、通过调整标定物或摄像机的方向，为标定物拍摄一些不同方向的照片。
3、从照片中提取棋盘格角点。
4、估算理想无畸变的情况下，五个内参和六个外参。
5、应用最小二乘法估算实际存在径向畸变下的畸变系数。
6、极大似然法，优化估计，提升估计精度。

我们需要标定手机摄像头的参数，就必须要预先拍摄一些含有棋盘标定板的相片，然后导入电脑进行分析，代码由 python 编写，可以在附录中查看

2.2 实验结果与分析

参考文献

[1] Haykin S. Adaptive Filter Theory[M]. 4th. Englewood Cliffs: Prentice Hall, 2002.

附录 A 算法

A.1 代码

源代码使用 listings 宏包，LMS 算法的 Verilog 模块端口声明如代码 A-1 所示。

代码 A-1 空时 LMS 算法 Verilog 模块端口声明

language

```
int main()
{

    std::string root_path = "E:\\Code\\heils_mobileface\\heils_mobileface\\heils_mobileface\\images\\lack";
    std::string suffix = ".jpg";

    std::vector<std::string> fns;
    glob(fns, root_path, suffix);
    //cout << pattern << endl;
    std::vector<cv::Mat> images;

    size_t face_num = 0;
    MtcnnDface facedetector = MtcnnDface();

    facedetector.initModel();
    for (int i = 0; i < fns.size(); i++) {
        cv::Mat cv_mat = cv::imread(fns[i]);
        ncnn::Mat ncnn_mat = ncnn::Mat::from_pixels(cv_mat.data, ncnn::Mat::PIXEL_BGR2RGB, cv_mat.cols, cv_mat.rows);

        std::vector<Bbox> finboxes;
        ptimer pt;
        //double start = static_cast<double>(cv::getTickCount());
        facedetector.detect(ncnn_mat, finboxes);
        //Sleep(1000);
        double inference = cv::getTickCount();
        std::cout << "boost timer: " << pt.elapsed() << "秒" << std::endl;
        //std::cout << "opencv timer: " << (inference - start) / cv::getTickFrequency() << "秒" << std::endl;

```

```
30          if (finboxes.size() != 0)
31              face_num++;
32          for (auto &box : finboxes)
33          {
34              cv::rectangle(cv_mat, cv::Rect(box.x1, box.y1, box.x2 - box.x1 + 1, box.y2 - box.y1 + 1),
                    cv::Scalar(0, 255, 255), 2);
35              for (int j = 0; j < 5; ++j)
36              {
37                  cv::circle(cv_mat, cvPoint(box.ppoint[j], box.ppoint[j + 5]), 2, CV_RGB(0, 255, 0),
                        CV_FILLED);
38              }
39          }
40          cv::imshow("show", cv_mat);
41          cv::waitKey(0);
42      }
43      std::cout<<"image count" << fns.size() <<"face num:"<< face_num << std::endl;
44
45      facedetector.releaseModel();
46
47      return 0;
48  }
```